Clean Copy of Replacement Pages for Brief Description of the Drawings

Brief Description of the Drawings

Fig. 1 is a block diagram of a computer system with a distributed processing system;

Figs. 2a-2b are block and flow diagrams of a distributed network management system;

Figs. 2c-2j are block and flow diagrams of distributed network management system clients and servers;

Figs. 3a-3b are a block diagram of a logical system model;

Figs. 3c and 3e-3g are flow diagrams depicting a software build process using a logical system model;

Fig. 3d is a flow diagram illustrating a method for allowing applications to view data within a database;

Fig. 3h is a flow diagram depicting a configuration process;

Figs. 3i and 3l are flow diagrams depicting template driven network services provisioning processes;

Figs. 3j-3k and 3m-3o are screen displays of an OSS client and various templates;

Figs. 4a-4z, 5a-5z, 6a-6p, 7a-7y, 8a-8e, 9a-9n, 10a-10i, 11a-11m, 11p-11q, 11u and 11z are screen displays of graphical user interfaces;

Figs. 11n-11o are tables representing data in a configuration database;

Figs. 11r-11t and 11v-11w are tables representing data in a network management system (NMS) database;

Fig. 11x is a block and flow diagram representing the creation of a user profile logical managed object including one or more groups;

Fig. 11y is a block and flow diagram of a network management system implementing user profiles and groups across multiple databases;

Figs. 12a and 13a are block and flow diagrams of a computer system incorporating a modular system architecture and illustrating a method for accomplishing hardware inventory and setup;

Figs. 12b-12c and 14a-14f are tables representing data in a configuration database;

Fig. 13b is a block and flow diagram of a computer system incorporating a modular system architecture and illustrating a method for configuring the computer system using a network management system;

Figs. 13c and 13d are block and flow diagrams of an accounting subsystem for pushing network device statistics to network management system software;

Fig. 15 is a block and flow diagram of a line card and a method for executing multiple instances of processes;

Figs. 16a-16b are flow diagrams illustrating a method for assigning logical names for inter-process communications;

Fig. 16c is a block and flow diagram of a computer system incorporating a modular system architecture and illustrating a method for using logical names for inter-process communications;

Fig. 16d is a chart representing a message format;

Figs. 17-19 are block and flow diagrams of a computer system incorporating a modular system architecture and illustrating methods for making configuration changes;

Fig. 20a is a block diagram of a packaging list;

Fig. 20b is a flow diagram of a software component signature generating process;

Figs. 20c and 20e are screen displays of graphical user interfaces;

Fig. 20d is a block and flow diagram of a network device incorporating a modular system architecture and illustrating a method for installing a new software release;

Fig. 21a is a block and flow diagram of a network device incorporating a modular system architecture and illustrating a method for upgrading software components;

Figs. 21b and 21g are tables representing data in a configuration database;

Figs. 21c-21f are screen displays of graphical user interfaces;

Fig. 22 is a block and flow diagram of a network device incorporating a modular system architecture and illustrating a method for upgrading a configuration database within the network device;

Fig. 23 is a block and flow diagram of a network device incorporating a modular system architecture and illustrating a method for upgrading software components;

Fig. 24 is a block diagram representing processes within separate protected memory blocks;

Fig. 25 is a block and flow diagram of a line card and a method for accomplishing vertical fault isolation;

Fig. 26 is a block and flow diagram of a computer system incorporating a hierarchical and configurable fault management system and illustrating a method for accomplishing fault escalation.

Fig. 27 is a block diagram of an application having multiple sub-processes;

Fig. 28 is a block diagram of a hierarchical fault descriptor;

Fig. 29 is a block and flow diagram of a computer system incorporating a distributed redundancy architecture and illustrating a method for accomplishing distributed software redundancy;

Fig. 30 is a table representing data in a configuration database;

Figs. 31a-31c, 32a-32c, 33a-33d and 34a-34b are block and flow diagrams of a computer system incorporating a distributed redundancy architecture and illustrating methods for accomplishing distributed redundancy and recovery after a failure;

Figs. 35a-35b are block diagrams of a network device;

Figs. 36a-36b are block diagrams of a portion of a data plane of a network device;

Fig. 37 is a block and flow diagram of a network device incorporating a policy provisioning manager;

Figs. 38 and 39 are tables representing data in a configuration database;

Fig. 40 is an isometric view of a network device;

Figs. 41a-41c are front, back and side block diagrams, respectively, of components and modules within the network device of Fig. 40;

Figs. 42a-42b are block diagrams of dual mid-planes;

Fig. 43 is a block diagram of two distributed switch fabrics and a central switch fabric;

Fig. 44 is a block diagram of the interconnections between switch fabric central timing subsystems and switch fabric local timing subsystems;

Figs. 45a-45b are block diagrams of a switch fabric central timing subsystem;

Fig. 46 is a state diagram of master / slave selection for switch fabric central timing subsystems;

Figs. 47a-47b are block diagrams of a switch fabric local timing subsystem;

Fig. 48 is a state diagram of reference signal selection for switch fabric local timing subsystems;

Fig. 49 is a block diagram of the interconnections between external central timing subsystems and external local timing subsystems;

Figs. 50a-50c are block diagrams of an external central timing subsystem;

Fig. 51 is a timing diagram of a first timing reference signal with an embedded second timing signal;

Fig. 52 is a block diagram of an embeddor circuit;

Fig. 53 is a block diagram of an extractor circuit;

Figs. 54a-54b are block diagrams of an external local timing subsystem;

Figs. 55a-55c are block diagrams of an external central timing subsystem;

Fig. 56 is a block diagram of a network device connected to test equipment through programmable physical layer test ports;

Fig. 57 is a block and flow diagram of a network device incorporating programmable physical layer test ports;

Fig. 58 is a block diagram of a test path table;

Fig. 59 is a block and flow diagram of a network management system incorporating proxies to improve NMS server scalability;

Figs. 60a-60n are tables representing data in a configuration database;

Fig. 61a is a block diagram representing a physical managed object;

Fig. 61b is a block diagram representing a proxy;

Fig. 62 is a screen display of a dialog box;

Figs. 63a-63b are block diagrams of a network device connected to an NMS;

Fig. 64 is a table representing data in an NMS database;

Fig. 65 is a block and flow diagram of a threshold management system;

Fig. 66a-66e are screen displays of a graphical user interface;

Fig. 67 is a screen display of a threshold dialog box;

Figs. 68, 69a-69b, 70a-70b and 71 are tables representing data in a configuration database;

Fig. 72a is a front, isometric view of a power distribution unit;

Fig. 72b is a rear, isometric view of the power distribution unit of Fig. 72a without a cover;

Fig. 73a is a rear, isometric view of a network device chassis including dual midplanes;

Figs. 73b-73c are enlarged views of portions of Fig. 73a;

Fig. 74 is a block and schematic diagram of a portion of a module including a power supply circuit;

Figs. 75, 76 and 79 are screen displays of a Virtual Connection Wizard;

Fig. 77 is a screen display of a VPI dialog box;

Fig. 78 is a screen display of a VPI/VCI dialog box;

Figs. 80 and 81 are block and flow diagrams of a common command interface;

Fig. 82 is a block and flow diagram of an application including a command API and a display API;

Fig. 83 is a block and flow diagram of an extended common command interface;

Fig. 84 is a block and flow diagram of a switch control plane within a network device including a distributed architecture;

Fig. 85 is a block and flow diagram of a switch subsystem;

Fig. 86 is a block and flow diagram of a redundant switch control planes within a network device including a distributed architecture; and

Fig. 87 is a block and flow diagram demonstrating distributed processors including multiple ports to each redundant switch control plane within a network device.

Clean Copy of Paragraph 2 on page 14

Referring to Fig. 2a, in the present invention, the NMS 60 includes one or more NMS client programs 850a-850n and one or more NMS server programs 851a-851n. The NMS client programs provide interfaces for network administrators. Through the NMS clients, the administrator may configure multiple network devices (e.g., computer system 10, Fig. 1; network device 540, Figs. 35a-35b). The NMS clients communicate with the NMS servers to provide the NMS servers with configuration requirements from the administrators. In addition, the NMS servers provide the NMS clients with network device management information, which the clients then make available to the administrators. "Pushing" data from a server to multiple clients synchronizes the clients with minimal polling. Reduced polling means less management traffic on the network and more device CPU cycles available for other management tasks. Communication between the NMS client and server is done via Remote Method Invocation (RMI) over Transmission Control Protocol (TCP), a reliable protocol that ensures no data loss.

Clean Copy of Paragraph 2 on page 15

The present invention also includes a configuration relational database 42 within each network device and an NMS relational database 61 external to the network device. The configuration database program may be executed by a centralized processor card or a processor on another card (e.g., 12, Fig. 1; 542, Figs. 35a-35b) within the network device, and the NMS database program may be executed by a processor within a separate computer system (e.g., 62, Fig. 13b). The NMS server stores data directly in the configuration database via JAVA Database Connectivity (JDBC) over TCP, and using JDBC over TCP, the configuration database, through active queries, automatically replicates any changes to NMS database 61. By using JDBC and a relational database, the NMS server is able to leverage database transactions, database views, database journaling and database backup technologies that help provide unprecedented system availability. Relational database technology also scales well as it has matured over many years. An active query is a mechanism that enables a client to post a blocked SQL query for asynchronous notification by the database when data changes are made after the blocked SQL query was made.

Clean Copy of Paragraph 2 on page 16

Referring again to Fig. 2a, for increased availability, the network device may include a backup configuration database 42' maintained by a separate, backup centralized processor card (e.g., 12, Fig. 1; 543, Figs. 35a-35b). Any changes to configuration database 42 are replicated to backup configuration database 42'. If the primary centralized processor card experiences a failure or error, the backup centralized processor card may be switched over to become the primary processor and configuration database 42' may be used to keep the network device operational. In addition, any changes to configuration database 42 may be written immediately to flash persistent memory 853 which may also be located on the primary centralized processor card or on another card, and similarly, any changes to backup configuration database 42' may be written immediately to flash persistent memory 853' which may also be located on the backup centralized processor card or another card. These flash-based configuration files protect against loss of data during power failures. In the unlikely event that all copies of the database within the network device are unusable, the data stored in the NMS database may be downloaded to the network device.

Clean Copy of Paragraph 3 beginning on page 16 and ending on page 17

Instead of having a single central processor card (e.g., 12, Fig. 1; 543, Figs. 35a-35b), the external control functions and the internal control functions may be separated onto different cards as described in U.S. Patent Application Serial Number 09/574,343, filed May 20, 2000 and entitled "Functional Separation of Internal and External Controls in Network Devices", which is hereby incorporated herein by reference. As shown in Figs. 41a and 41b, the chassis may support internal control (IC) processor cards 542a and 543a and external control (EC) processor cards 542b and 543b. In this embodiment, configuration database 42 may be maintained by a processor on internal control processor card 542a and configuration database 42' may be maintained by a processor on internal control processor card 543a, and persistent memory 853 may be located on external control processor card 542b and persistent memory 853' may be located on external control processor card 543b. This increases inter-card communication but also provides increased fault tolerance.

Clean Copy of Paragraph 2 on Page 26

Referring to Figs. 3a-3b, a logical system model 280 is created using the object modeling notation and a model generation tool, for example, Rational Rose 2000 Modeler Edition available from Rational Software Corporation in Lexington, Massachusetts. A managed device 282 represents the top level system connected to models representing both hardware 284 and data objects used by software applications 286. Hardware model 284 includes models representing specific pieces of hardware, for example, chassis 288, shelf 290, slot 292 and printed circuit board 294. The logical model is capable of showing containment, that is, typically, there are many shelves per chassis (1:N), many slots per shelf (1:N) and one board per slot (1:1). Shelf 290 is a parent class generalizing multiple shelf models, including various functional shelves 296a-296n as well as one or more system shelves, for example, for fans 298 and power 300. Board 294 is also a parent class having multiple board models, including various functional boards without external physical ports 302a-302n (e.g., central processor 12, Fig. 1; 542-543, Fig. 35a; and switch fabric cards, Figs. 35a-35b) and various functional boards 304a-304n (e.g., cross connection cards 562a-562b and forwarding cards 546a-546e, Fig. 35a) that connect to boards 306 with external physical ports (e.g., universal port cards 554a-554h, Fig. 35a). Hardware model 284 also includes an external physical port model 308. Port model 308 is coupled to one or more specific port models, for example, synchronous optical network (SONET) protocol port 310, and a physical service endpoint model 312.

Clean Copy of Paragraph 3 beginning on page 26 and ending on Page 27

Hardware model 284 includes models for all hardware that may be available on computer system 10 (Fig. 1) / network device 540 (Figs. 35a-35b) whether a particular computer system / network device uses all the available hardware or not. The model defines the metadata for the system whereas the presence of hardware in an actual network device is represented in instance data. All shelves and slots may not be populated. In addition, there may be multiple chassis. It should be understood that SONET port 310 is an example of one type of port that may be supported by computer system 10. A model is created for each type of port available on computer system 10, including, for example, Ethernet, Dense Wavelength Division Multiplexing (DWDM) or Digital Signal, Level 3 (DS3). The NMS (described below) uses the hardware model and instance data to display a graphical picture of computer system 10 / network device 540 to a user.

Clean Copy of Paragraph 1 on Page 29

Referring to Fig. 3c, logical model 280 is used as input to a code generation system 336. The code generation system creates a view identification (id) and an application programming interface (API) 338 for each process that requires configuration data. For example, a view id and an API may be created for each ATM application 339a-339n, each SONET application 340a-340n, each MPLS application 342a-342n and each IP application 341a-341n. In addition, a view id and API is also created for each device driver process, for example, device drivers 343a-343n, and for modular system services (MSS) 345a-345n (described below), for example, a Master Control Driver (MCD), a System Resiliency Manager (SRM), and a Software Management System (SMS). The code generation system provides data consistency across processes, centralized tuning and an abstraction of embedded configuration and NMS databases (described below) ensuring that changes to their database schema (i.e., configuration tables and relationships) do not affect existing processes.

Clean Copy of Paragraph 1 on Page 30

Referring to Fig. 3d, applications 352a-352n (e.g., SONET driver 863, SONET application 860, MSS 866, etc.) each have an associated view 354a-354n of configuration database 42. The views may be similar allowing each application to view similar data within configuration database 42. For example, each application may be ATM version 1.0 and each view may be ATM view version 1.3. Instead, the applications and views may be different versions. For example, application 352a may be ATM version 1.0 and view 354a may be ATM view version 1.3 while application 352b is ATM version 1.7 and view 354b is ATM view version 1.5. A later version, for example, ATM version 1.7, of the same application may represent an upgrade of that application and its corresponding view allows the upgraded application access only to data relevant to the upgraded version and not data relevant to the older version. If the upgraded version of the application uses the same configuration data as an older version, then the view version may be the same for both applications. In addition, application 352n may represent a completely different type of application, for example, MPLS, and view 354n allows it to have access to data relevant to MPLS and not ATM or any other application. Consequently, through the use of database views, different versions of the same software applications and different types of software applications may be executed on computer system 10 simultaneously.

Clean Copy of Paragraph 3 beginning on Page 30 and ending on Page 31

Referring again to Fig. 3c, logical model 280 may be changed (280') to include models representing the new software and/or hardware. Code generation system 336 then uses new logical model 280' to re-generate view ids and APIs 338' for each application, including, for example, ATM version two 360 and device driver 362, and DDL files 344' and 348'. The new application(s) and/or device driver(s) processes then bind to the new view ids and APIs. A copy of the new application(s) and/or device driver process as well as the new DDL files and any new hardware are sent to the user of computer system 10. The user can then download the new software and plug the new hardware into computer system 10. The upgrade process is described in more detail below. Similarly, if models are upgraded / modified to reflect upgrades / modifications to software or hardware, then the new logical model is provided to the code generation system which re-generates view ids and APIs for each process / program / application. Again, the new applications are linked with the new view ids and APIs and the new applications and/or hardware are provided to the user.

Clean Copy of Paragraph 2 on Page 31

Again referring to Fig. 3c, the code generation system also creates NMS JAVA interfaces 347 and persistent layer metadata 349. The JAVA interfaces are JAVA class files including get and put methods corresponding to attributes within the logical model, and as described below, the NMS servers use the NMS JAVA interfaces to construct models of each particular network device to which they are connected. Also described below, the NMS servers use the persistent layer metadata as well as run time configuration data to generate SQL configuration commands for use by the configuration database.

Clean Copy of Paragraph 3 on Page 31

Prior to shipping computer system 10 to customers, a software build process is initiated to establish the software architecture and processes. The code generation system is the first part of this process. Following the execution of the code generation system, each process when pulled into the build process links the associated view id and API into its image. For example, referring to Fig. 3e, to build a SONET application, source files, for example, a main application file 859a, a performance monitoring file 859b and an alarm monitoring file 859c, written in, for example, the C programming language (.c) are compiled into object code files (.o) 859a', 859b' and 859c'. Alternatively, the source files may be written in other programming languages, for example, JAVA (.java) or C++ (.cpp). The object files are then linked along with view ids and APIs from the code generation system corresponding to the SONET application, for example, SONET API 340a. The SONET API may be a library (.a) of many object files. Linking these files generates the SONET Application executable file (.exe) 860.

Clean Copy of Paragraph 2 on Page 32

Referring to Fig. 3f, each of the executable files for use by the network device / computer system are then provided to a kit builder 861. For example, several SONET executable files (e.g., 860, 863), ATM executable files (e.g., 864a-864n), MPLS executable files (e.g., 865a-865n), MSS executable files 866a-866n, MKI executable 873a-873n files for each board and a DDL configuration database executable file 867 may be provided to kit builder 861. The OSE operating system expects executable load modules to be in a format referred to as Executable & Linkable Format (.elf). Alternatively, the DDL configuration database executable file may be executed and some data placed in the database prior to supplying the DDL file to the kit builder. The kit builder creates a computer system / network device installation kit 862 that is shipped to the customer with the computer system / network device or, later, alone after modifications and upgrades are made. To save space, the kit builder may compress each of the files included in the Installation Kit (i.e., .exe.gz, .elf.gz), and when the files are later loaded in the network device, they are de-compressed.

Clean Copy of Paragraph 3 beginning on Page 32 and ending on Page 33

Referring to Fig. 3g, similarly, each of the executable files for the NMS is provided separately to the kit builder. For example, a DDL NMS database executable file 868, an NMS JAVA interfaces executable file 869, a persistent layer metadata executable file 870, an NMS server 885 and an NMS client 886 may be provided to kit builder 861. The kit builder creates an NMS installation kit 871 that is shipped to the customer for installation on a separate computer 62 (Fig. 13b). In addition, new versions of the NMS installation kit may be sent to customers later after upgrades / modifications are made. When installing the NMS, the customer / network administrator may choose to distribute the various NMS processes as described above. Alternatively, one or more of the NMS programs, for example, the NMS JAVA interfaces and Persistent layer metadata executable files may be part of the network device installation kit and later passed from the network device to the NMS server, or part of both the network device installation kit and the NMS installation kit.

Clean Copy of Paragraph 3 on Page 33

As described above, logical model 280 (Fig. 3c) may be provided as an input to code generation system 336 in order to generate database views and APIs for NMS programs and network device programs to synchronize the integration interfaces between those programs. Where a telecommunications network includes multiple similar network devices, the same installation kit may be used to install software on each network device to provide synchronization across the network. Typically, however, networks include multiple different network devices as well as multiple similar network devices. A logical model may be created for each different type of network device and a different installation kit may be implemented on each different type of network device.

Clean Copy of Paragraph 4 on Page 33

Instead, of providing a logical model (e.g., 280, Fig. 3c) that represents a single network device, a logical model may be provided that represents multiple different managed devices – that is, multiple network devices and the relationship between the network devices. Alternatively, multiple logical models 280 and 887a-887n – representing multiple network devices -- may be provided, including relationships with other logical models. In either case, providing multiple logical models or one logical model representing multiple network devices and their relationships as an input(s) to the code generation system allows for synchronization of NMS programs and network device programs (e.g., 901a-901n) across an entire network. The code generation system in combination with one or more logical models provides a powerful tool for synchronizing distributed telecommunication network applications.

Clean Copy of Paragraph 4 on Page 34

Configuration:

Once the network device programs have been installed on network device 540 (Figs. 35a-35b), and the NMS programs have been installed on one or more computers (e.g., 62), the network administrator may configure the network device / provision services within the network device. Hereinafter, the term "configure" includes "provisioning services". Referring to Fig. 4a, the NMS client displays a graphical user interface (GUI) 895 to the administrator including a navigation tree / menu 898. Selecting a branch of the navigation tree causes the NMS client to display information corresponding to that branch. For example, selecting Devices branch 898a within the tree causes the NMS client to display a list 898b of IP addresses and/or domain name server (DNS) names corresponding to network devices that may be managed by the administrator. The list corresponds to a profile associated with the administrator's user name and password. Profiles are described in detail below.

Clean Copy of Paragraph 2 on Page 35

To configure a network device, the administrator begins by selecting (step 874, Fig. 3h) a particular network device to configure, for example, the network device corresponding to IP address 192.168.9.202 (Fig. 4f). The NMS client then informs (step 875, Fig. 3h) an NMS server of the particular network device to be configured. Since many NMS clients may connect to the same NMS server, the NMS server first checks its local cache to determine if it is already managing the network device for another NMS client. If so, the NMS server sends data from the cache to the NMS client. If not, the NMS server using JDBC connects to the network device and reads the data / object structure for the physical aspects of the device from the configuration database within the network device into its local cache and uses that information with the JAVA interfaces to construct (step 876) a model of the network device. The server provides (step 877) this information to the client, which displays (step 878) a graphical representation 896a (Fig. 4f) of the network device to the administrator indicating the hardware and services available in the selected network device and the current configuration and currently provisioned services. Configuration changes received by an NMS server – from either an NMS client or directly from the network device's configuration database when changes are made through the network device's CLI interface -- are sent by the NMS server to any other NMS clients connected to that server and managing the same network device. This provides scalability, since the device is not burdened with multiple clients subscribing for traps, and ensures each NMS client provides an accurate view of the network device.

Clean Copy of Paragraph 1 on Page 36

Referring to Figs. 4f-4l, graphical representation 896a (i.e., device view, device mimic) in graphic window 896b may include many views of the network device. For example, device mimic 896a is shown in Fig. 4f displaying a front view of the components in the upper portion of network device 540 (Figs. 35a-35b). The administrator may use scroll bar 926a to scroll down and view lower portions of the front of the network device as shown in Fig. 4g. The administrator may also use image scale button 926b to change the size of graphic 896a. For example, the administrator may shrink the network device image to allow more of the device image to be visible in graphic window 896b, as shown in Fig. 4h. This view corresponds to the block diagram of network device 540 shown in Fig. 41a. For instance, upper fan tray 634 and middle fan trays 630 and 632 are shown. In addition, forwarding cards (e.g., 546a and 548e), cross-connection cards (e.g., 562a, 562b, 564b, 566a, 568b), and external processor control cards (e.g., 542b and 543b) are shown.

Clean Copy of Paragraph 3 beginning on page 37 and ending on Page 38

Device mimic 896a is shown in Fig. 4i displaying a back view of the components in the upper portion of network device 540 (Figs. 35a-35b). Again the administrator may use scroll bar 926a and/or image scale button 926b to view lower portions (Figs. 4j and 4k) of the back of the network device or more of the network device by shrinking the graphic (Fig. 4l). These views correspond to the block diagram of network device 540 shown in Fig. 41b. For example, upper fan tray 628 (Fig. 4i), management interface (MI) card 621 (Fig. 4i) and lower fan tray 626 (Fig. 4k) are shown. In addition, universal port cards (e.g., 556h, 554a and 560h, Fig. 4l), switch fabric cards (e.g., 570a and 570b) and internal processor control cards (e.g., 542a and 543a) are also shown. Again, graphic 896a may use a visual indicator to clearly show whether a card is present in a slot or whether the slot is empty. In this example, the visual indicator for universal port cards is the display of the ports available on each card. For example, universal port card 554a is present as indicated by the graphical representation of ports (e.g., 930, Fig. 4l) available on that card, while universal port card 558a (Fig. 41b) is not present as indicated by a blank slot 931.

Clean Copy of Paragraph 2 on Page 41

The System tab data as well as the Modules tab, Ports tab and SONET Interface tab data all represent physical aspects of the network device. The remaining tabs, including SONET Paths tab 942 (Fig. 4w), ATM Interfaces tab 946, Virtual ATM Interfaces tab 947 and Virtual Connections tab 948, display configuration details and, thus, display no data until the device is configured. In addition, these configuration tabs 942, 946-948 are dialog chained together with wizard-like properties to guide an administrator through configuration details. Through these tabs within the GUI (i.e., graphical context), therefore, the administrator then makes (step 879, Fig. 3h) configuration selections. For example, to configure a SONET path, the administrator may begin by selecting a port (e.g., 939a on card 556e, Fig. 5a) within device mimic 896a and clicking the right mouse button (i.e., context sensitive) to cause a pop-up menu 943 to be displayed listing available port configuration options. The administrator may then select the "Configure SONET Paths" option, which causes the GUI to display a SONET Path configuration wizard 944 (Fig. 5b).

Clean Copy of Paragraph 3 beginning on Page 43 and ending on Page 44

Once the administrator selects the OK button, the NMS client validates the parameters as far as possible within the client's view of the device and passes (step 880, Fig. 3h) this run time / instance configuration data, including all configured SONET path parameters, to the NMS server. The NMS server validates (step 881) the data received based on its view of the world and if not correct, sends an error message to the NMS client, which notifies the administrator. Thus, the NMS server re-validates all data from the NMS clients to ensure that it is consistent with changes made by any other NMS client or by an administrator using the network device's CLI. After a successful NMS server validation, the Persistent layer software within the server uses this data to generate (step 882) SQL commands, which the server sends to the configuration database software executing on the network device. This is referred to as "persisting" the configuration change. Receipt of the SQL commands triggers a validation of the data within the network device as well. If the validation is not successful, then the network device sends an error message to the NMS server, and the NMS server sends an error message to the NMS client, which displays the error to the administrator. If the validation is successful, the configuration database software then executes (step 883) the SQL commands to fill in or change the appropriate configuration tables.

Clean Copy of Paragraph 3 beginning on page 80 and ending on page 81

Custom Object Collections:

As described above with respect to FCAPS management, a network device (e.g., 10, Fig. 1 and 540, Figs. 35a-35b) may include a large number (e.g., millions) of configurable / manageable objects such as modules, ports, paths, connections, etc. To provide flexibility and scalability, the network management system (NMS) allows users to create custom object collections. Thus, even though a network device or multiple network devices in a telecommunication network may include millions of objects, a network manager may create a collection and add only objects of interest to that collection. The objects may be of a similar or different type and may correspond to the same or different network devices. The network manager may also add and remove objects from existing collections, create additional new collections and remove existing collections. The network manager may then view the various objects in each collection. In addition, the collections are linked to the NMS graphical user interface (GUI), such that changes to objects in either are updated in the other. Custom object collections provide scalability and flexibility. In addition, custom object collections may be tied to user profiles to limit access. For example, a customer may be limited to viewing only the collections of objects related to their account. Similarly, a network manager may be limited to viewing only those collections of objects for which they have authority.

Clean Copy of Paragraph 3 on Page 85

To change the parameters in the network administrator's profile or any other existing profile, including a copied profile, the user double clicks on one of the profiles 904. To add a new profile, the user clicks on an Add button 905. In either case, the NMS client displays a profile dialog box 907 (Figs. 11b-11c) on the screen. Through the profile dialog box, a user's user name 908a, password 908b and confirmed password 908c may be added or changed. The confirm password field is used to assure that the password was entered properly in the password field. The password and confirmed password may be encrypted strings used for user authentication. These fields will be displayed as asterisks on the screen. Once added, a user simply logs on to an NMS client with this user name and password and the NMS client displays the GUI in accordance with the other parameters of this profile.

Clean Copy of Paragraph 2 on Page 87

As described below, the information provided in a user profile is stored in tables within the NMS database, and when a user logs onto the network through an NMS client, the NMS client connects to an NMS server that retrieves the user's profile information and sends the information to the NMS client. The NMS client automatically saves the NMS server primary and secondary IP addresses and port numbers from the user's profile to a team session file associated with the user's username and password in a memory 986 (Fig 11y) local to the NMS client. If the user logs into an NMS client through a web browser, then the NMS client may save the NMS server primary and secondary IP addresses and port numbers to a cookie that is then stored in the user's local hard drive. The next time the user logs in to the NMS client, the NMS client uses the IP addresses and port numbers stored in the team session file or cookie to connect to the appropriate NMS server.

Clean Copy of Paragraph 3 on Page 87

The first time a user accesses an NMS client, however, no team session file or cookie will be available. Consequently, during the initial access of the NMS client, the NMS client may use a default IP address to connect with an NMS server or a pop-up menu 1034 (Fig. 11z) may be displayed in which the user may type in the IP address in a field 1034a of the NMS server they want the NMS client to use or select an IP address from a pop-up menu that appears when a dropdown button 1034b is selected.

Clean Copy of Paragraph 2 on Page 88

Referring again to Figs. 11b-11c, additional fields may be added to device list 908g to provide more information. For example, a read field 908p may be used to indicate the SNMP community string to be used to allow the NMS server to communicate with the network device over SNMP. The SNMP connection may be used to retrieve statistical data and device status from the network device. In addition, a read/write field 908q may be used to indicate an SNMP community string to allow the NMS server to configure the network device and/or provision services. The profile may also include a retry field 908r and a timeout field 908s to provide SNMP retry and timeout values. Many different fields may be provided in a profile.

Clean Copy of Paragraph 3 on Page 88

Instead of providing all the parameters and fields in a single profile dialog box, they may be separated into a variety of a tabbed dialog boxes (Figs. 11d-11g). The tabbed dialog boxes may provide better scalability and flexibility for future needs.

Clean Copy of Paragraph 3 on Page 89

Referring to Fig. 11h, a user preference dialog box 909 may be used to customize the GUI into a presentation format that is most efficient or easy for a user to work with. For example, show flags (i.e., attributes) may be used to add tool tips (flag 910a), add horizontal grid lines on tables (flag 910b), add vertical grid lines on tables (flag 910c) and add bookmarks / short cuts (e.g., create a short cut to a PVC dialog box). Look and feel flags may also be used to make the GUI appear as a JAVA GUI would appear (flag 911a) or as a native application, for example, Windows, Windows/NT or Motif, GUI would appear (flag 911b).

Clean Copy of Paragraph 4 beginning on Page 89 and ending on Page 90

As an alternative to providing a Group Name 908f (Figs. 11b-11c) or a Customer Name (Fig. 11d), when a profile is created or changed the administrator or provisioner may double click the left mouse button on a network device (e.g., 192.168.9.202, Figs. 11b-11c or 11g) in the device list to cause a pop-up menu 1000 (Figs. 11i-11j) to be displayed. The pop-up menu provides a list 1000a of available groups corresponding to the selected network device, and the administrator or provisioner may select one or more groups (e.g., Walmart-East, Walmart-West) from the list for which the user corresponding to profile will be authorized to access.

Clean Copy of Paragraph 3 beginning on Page 90 and ending on Page 91

When an administrator or a provisioner configures a network device resource, they may assign that resource to a particular group. For example, when an administrator or provisioner configures one or more SONET paths, they may assign each SONET path to a particular group. Referring to Fig. 11k-11m, within a SONET Path configuration wizard 1002, an administrator or provisioner may select a SONET Path within the SONET path table 1002a and type in a group name in field 1002b or select a group name from a pop-up menu displayed when dropdown button 1002c is selected. When the administrator / provisioner selects OK button 1002d or Modify button 1002e, the NMS client sends the SONET path data to the NMS server. The NMS server uses this data to fill in a SONET path table (e.g., 600', Figs. 11y and 60g) in configuration database 42. A new row is added to the SONET path table for each newly configured SONET path, and data in existing rows are modified for modified SONET paths.

Clean Copy of Paragraph 2 on Page 91

In addition, the NMS server searches a Managed Resource Group table 1008 (Figs. 11n and 11y) within the configuration database for a match with each assigned group name. If no match is found for a group name, indicating the group name represents a new group, then the NMS server adds a row to the Managed Resource Group table, and the NMS server assigns the group an LID (e.g., 1145) and inserts the LID into an LID column 1008a. The NMS server also inserts the Managed Device PID (e.g., 1) from column 983b in Managed Device table 983 (Figs. 11y and 60a) in the configuration database into a column 1008b and inserts the group name in column 1008c.

Clean Copy of Paragraph 3 on Page 91

The NMS server also uses the SONET path data from the NMS client to add a row in a Managed Resource Table 1007 (Figs. 11o and 11y) in configuration database 42 for each newly configured SONET path or to modify data in existing rows for modified SONET paths. The NMS server assigns an LID (e.g., 4443) to each row and inserts the assigned LID into a column 1007a. The NMS server then inserts the assigned SONET path LID (e.g., 901) from Path LID column 600a (Fig. 60g) in the SONET path table into a Resource LID column 1007b. The NMS server also inserts the assigned group LID (e.g., 1145) from column 1008a in Managed Resource Group table 1008 (Fig. 11n) into a managed resource group LID column 1007c.

Clean Copy of Paragraph 4 beginning on Page 91 and ending on Page 92

Just as each SONET path may be assigned to a group, each other type of configured resource / manageable entity within the network device may be assigned to a group. For example, when an administrator or provisioner configures a virtual ATM (VATM) interface, they may also assign the VATM interface to a group. Referring to Fig. 11p, within an Add V-ATM Interface dialog box 1004, an administrator or provisioner may type in a group name in a field 1004a or select a group name from a pop-up menu displayed when expansion button 1004b is selected. As another example, when an administrator or provisioner configures an ATM PVC, they may assign the ATM PVC to a particular group. Referring to Fig. 11q, in a virtual connection wizard 1006, the administrator or provisioner may assign an ATM PVC to a group by typing in a group name in a field 1006a or by selecting a group name from a pop-up menu displayed when expansion button (e.g., Group List) 1006b is selected. Again, when the administrator or provisioner selects OK button 1004c (Fig. 11p) or Finish button 1006c (Fig. 11q), the NMS client sends the relevant data to the NMS server. The NMS server updates Virtual ATM Interface table 993 (Fig. 60j), a Virtual Connection table 994 (Fig. 60k), Virtual Link table 995 (Fig. 60L) and Cross-Connect table 996 (Fig. 60m), as described below, and similar to the actions taken for the configured SONET paths, the NMS server adds a row to Managed Resource Group table 1008 (Fig. 11n) for each new group and a row to Managed Resource table 1007 (Fig. 11o) for each new managed resource – that is, for each new VATM interface and for each new ATM PVC. This same process may be used to add any manageable entity to a group.

Clean Copy of Paragraph 2 beginning on Page 92 and ending on Page 93

Instead of using a Managed Resource Group table and a Managed Resource table, the configured network device resource tables (e.g., SONET path table, Virtual ATM IF table, etc.) could include a group name field. However, the Managed Resource Group adds a layer of abstraction, which may allow each configured resource to belong to multiple groups. Moreover, the Managed Resource table provides scalability and modularity by not being tied to a particular resource type. That is, the Managed Resource table will include a row for each different type of configured resource and if the network device is upgraded to include new types of configurable resources, they too may be added to the Managed Resource table without having to upgrade other processes. If each configurable resource is limited to belonging to only one group, then the Managed Resource Table 1007 (Fig. 11o) may include only Resource LID 1007b and not LID 1007a.

Clean Copy of Paragraph 2 on Page 93

Referring again to Figs. 11b-11h, after adding or changing a user profile, the administrator or provisioner selects OK button 908t. Selection of the OK button causes the NMS client (e.g., NMS client 850a, Fig. 11y) to send the information provided in the dialog box (or boxes) to an NMS server (e.g., NMS server 851a), and the NMS server uses the received information to update various tables in NMS database 61. In one embodiment, for a newly added user, the NMS server assigns a unique logical identification number (LID) to the user and adds a new row in a User table 1010 (Figs. 11r-11y) in the NMS database including the assigned LID 1010a and the username 1010b, password 1010c and group access level 1010d provided by the NMS client. For example, the NMS server may add a new row 1010e including an assigned user LID of 2012, a username of Dave, a password of Marble and a group access level of provisioner.

Clean Copy of Paragraph 3 on Page 93

The NMS server also adds a row to a User Managed Device table 1012 (Figs. 11s and 11y) for each network device listed in the user profile. For each row, the NMS server assigns a user managed device LID (e.g., 7892) and inserts it in an LID column 1012a. The NMS server also inserts a user LID 1012b, a host LID 1012c, a retry value 1012d and a timeout value 1012e. The inserted retry and timeout values are from the user profile information sent from the NMS client. The user LID 1012b includes the previously assigned user LID (e.g., 2012) from column 1010a of User Table 1010. The host LID is retrieved from an Administration Managed Device table 1014 (Figs. 11t and 11y).

Clean Copy of Paragraph 4 beginning on Page 93 and ending on Page 95

The Administration Managed Device table includes a row for each network device (i.e., managed device) in the telecommunications network. To add a network device to the network, an administrator selects an Add Device option in a pop-up menu 898c (Fig. 6a) in GUI 895 to cause dialog box 1013 (Fig. 11u) to be displayed. The administrator enters the intended IP address or DNS name (e.g., 192.168.9.202) of the new network device into a device host field 1013a and may also enter a device port (e.g., 1521) into a device port field 1013b. The administrator also adds SNMP retry 1013c and timeout 1013d values, which may be overridden later by values supplied within each user profile. In addition, the administrator adds a password for each user access level. In one embodiment, the administrator adds an administrator password 1013e, a provisioner password 1013f and a viewer password 1013g for the managed device.

Clean Copy of Paragraph 4 beginning on Page 94 and ending on Page 95

When the NMS server adds a new row to the User Managed Device table 1012 (Fig. 11s), corresponding to a managed device in a user profile, the NMS server searches column 1014b in the Administration Managed Device table 1014 for a host address matching the IP address (e.g., 192.168.9.202) provided in the user profile information sent from the NMS client. When a match is found, the NMS server retrieves the host LID (e.g., 9046) from column 1014a and inserts it in host LID column 1012c in the User Managed Device table.

Clean Copy of Paragraph 2 on Page 95

After receiving user profile information from an NMS client, the NMS server also updates a User Resource Group Map table 1016 (Figs. 11v and 11y) in NMS database 61. For each group identified in the user profile information – one or more groups may be selected in each Group List dialog box 1000 associated with each network device in the user profile – the NMS server adds a row to the User Resource Group Map table. The NMS server assigns an LID (e.g., 8086) for each row and inserts the LID in a column 1016a. The NMS server then inserts the User LID (e.g., 2012) into User LID column 1016b from User table 1010 column 1010a corresponding to the user profile. In addition, the NMS server inserts a User Resource Group LID into column 1016c.

Clean Copy of Paragraph 3 on Page 95

For each group name received by the NMS server, the NMS server searches a User Resource Group table 1018 (Figs. 11w and 11y), group name column 1018c, for a match. If a match is not found, then the group is a new group, and the NMS server adds a row to the User Resource Group table. The NMS server assigns an LID (e.g., 1024) to each row and inserts the assigned LID into an LID column 1018a. This User Resource Group LID is also added to column 1016c in the User Resource Group Map table 1016 (Fig. 11v). Within the User Resource Group table 1018 (Fig. 11w), the NMS server also inserts the network device's host LID in a column 1018b from Administration Managed Device table 1014 (Fig. 11t), column 1014a, and the NMS server inserts the group name (e.g., Walmart-East) in column 1018c. Through the group name, the User Resource Group table in the NMS database provides for dynamic binding with the Managed Resource Group table 1008 (Fig. 11n) in the configuration database, as described below.

Clean Copy of Paragraph 4 beginning on Page 95 and ending on Page 96

After a user's profile is created, the user may log in through an NMS client (e.g., 850a, Fig. 11y) by typing in their username and password. The NMS client then sends the username and password to an NMS server (e.g., 851a), and in response, the NMS server sends a query to NMS database 61 to search User table 1010 (Fig. 11r) column 1010b for a username matching the username provided by the NMS client. If the username is not found, then the user is denied access. If the username is found, then, for additional security, the NMS server may compare the password provided by the NMS client to the password stored in column 1010c of the User table. If the passwords do not match, then the user is denied access. If the passwords match, then the NMS server creates a user profile logical managed object (LMO).

Clean Copy of Paragraph 2 on Page 96

In one embodiment, the user profile LMO is a JAVA object and a JAVA persistence layer within the NMS server creates the user profile LMO. For each persistent JAVA class / object, metadata is stored in a class table 1020 (Fig. 11y) within the NMS database. Thus, the JAVA persistence layer within the NMS server begins by retrieving metadata from the class table in the NMS database corresponding to the user profile LMO. The metadata may include simple attributes and association attributes.

Clean Copy of Paragraph 3 on Page 96

Referring to Fig. 11x, the metadata for a user profile LMO 1022 includes three simple attributes – username 1022a, password 1022b and group access level 1022c – and two association attributes – resource group maps 1022d and managed devices 1022e. The NMS server inserts the username (e.g., Dave), password (e.g., Marble) and group access level (e.g., provisioner) retrieved from the User table 1010 into the user profile LMO 1024 (Fig. 11y) being created. The managed devices association attribute 1022e causes the NMS server to create a user managed device properties LMO 1026 for each network device in the user's profile.

Clean Copy of Paragraph 4 beginning on Page 96 and ending on Page 97

The NMS server first retrieves metadata from class table 1020 associated with the user managed device properties LMO 1026. The metadata includes two simple attributes (retry 1026b and timeout 1026c) and one association attribute (managed device 1026a). The metadata causes the NMS server to search User Managed Device table 1012 (Fig. 11s) column 1012b for a user LID (e.g., 2012) corresponding to the user LID in column 1010a (Fig. 11r) of User table 1010 in a row 1010e associated with the username and password received from the NMS client. For each row in the User Managed Device table having the matching user LID (e.g., 2012), the NMS server creates a user managed device properties LMO 1026 and inserts the retry value from column 1012d as the retry simple attribute 1026b and the timeout value from column 1012e as the timeout simple attribute 1026c.

Clean Copy of Paragraph 3 on Page 97

The NMS server uses the host LID (e.g., 9046) from column 1012c in the User Managed Device table (Fig. 11s) as a primary key to locate the row (e.g., 1014c, Fig. 11t) in the Administration Managed Device table 1014 corresponding to the network device. The NMS server uses the data in this table row to insert values for the simple attributes in the Administration Managed Device LMO 1028. For example, a host address of 192.168.9.202 and a port address of 1521 may be inserted. The NMS server also selects a password corresponding to the user's group access level. For instance, if the user's group access level is provisioner, then the NMS server inserts the provisioner password of, for example, team2, from column 1014d into the Administration Managed Device LMO.

Clean Copy of Paragraph 1 on Page 98

The resource group maps association attribute 1022d (Fig. 11x) within user LMO 1022 causes the NMS server to create a user resource group map LMO 1030 for each group in the user's profile. The user resource group map LMO 1030 includes one simple attribute -- user profile 1030a -- and one association attribute – user resource group 1030b. The NMS server inserts the user LID (e.g., 2012) corresponding to the user LID in column 1010a (Fig. 11r) in User table 1010 associated with the username, password and group access level received from the NMS client.

Clean Copy of Paragraph 2 on Page 98

In response to user resource group associated attribute 1030b, the NMS server creates a User Resource Group LMO 1032. The NMS server begins by retrieving metadata from class table 1020 corresponding to the User Resource Group LMO. The metadata includes three simple attributes: host address 1032a, port address 1032b and group name 1032c. The NMS server searches User Resource Group Map table 1016 (Fig. 11v) for the user LID (e.g., 2012) corresponding to the username and password received from the NMS client. The NMS server then uses the corresponding user resource group LID (e.g., 1024) from column 1016c as a primary key to locate a row (e.g., 1018d, Fig. 11w) in User Resource Group table 1018. The NMS server inserts the group name (e.g., Walmart-East) from the located row in User Resource Group table 1018 as simple attribute 1032c in user resource group LMO 1032. The NMS server then uses the host LID (e.g., 9046) from the located row to search column 1014a in the Administration Managed Device table 1014 (Fig. 11t) for a match. Once a match is found, the NMS server uses data in the located row (e.g., 1014c) to insert the host address (e.g., 192.168.9.202) from column 1014b as simple attribute 1032a and the port address (e.g., 1521) from column 1014e as simple attribute 1032b in user resource group LMO 1032. The NMS server then inserts the user resource group LMO 1032 into the user resource group map LMO 1030, and the NMS server inserts each of the user resource group map LMOs 1030 into the user profile LMO 1022. Thus, the data (e.g., host and port address and group name) required to locate each group included in the user profile is inserted within user profile LMO 1022.

Clean Copy of Paragraph 2 on Page 100

For example, if a user selects SONET Paths tab 942 (Fig. 5q), then the NMS server searches the user profile LMO for all group names corresponding to the selected network device (e.g., Walmart-East) or the NMS client provides all group names (e.g., Walmart-East) corresponding to the selected network device to the NMS server as part of the "Get SONET paths" transaction. The NMS server then dynamically issues a where clause such as "where SONET path is in group Walmart-East". This causes group name column 1008c in the Managed Resource Group table 1008 (Fig. 11n) in the network device's configuration database 42 to be searched for a match with the group name of Walmart-East. Additional where clauses may be dynamically issued corresponding to other group names found in the user profile LMO. If no match is found for a group name in column 1008c, then the NMS server simply returns an empty set to the NMS client. If a match is found for a group name (e.g., Walmart-East), then the NMS server retrieves the managed resource group LID (e.g., 1145) from column 1008a in the same row (e.g., row 1008d) as the matching group name.

Clean Copy of Paragraph 3 beginning on Page 100 and ending on Page 101

The NMS server then searches column 1007c in the Managed Resource table 1007 (Fig. 11o) for one or more matches with the retrieved managed resource group LID (e.g., 1145). As described above, the Managed Resource Table includes one row for each configured network device resource in a particular group. For each match found for the retrieved managed resource group LID (e.g., 1145), the NMS server uses the resource LID (e.g., 901) from column 1007b as a primary key to a row in a table including the data corresponding to the configured resource. In this example, a resource LID of 901 corresponds to a row in SONET Path Table 600' (Fig. 60g). Since the user selected the SONET Paths tab, the NMS server retrieves the data in the corresponding row and sends it to the NMS client. The NMS client uses the data to update graphical user interface (GUI) tables 985 in local memory 986, which causes GUI 895 to display the SONET path to the user. Other SONET paths may also be included in the group Walmart-East, and those would be similarly located and retrieved by the NMS server and sent to the NMS client for display to the user.

Clean Copy of Paragraph 3 beginning on Page 101 and ending on Page 102

When data is stored in tables within the same database, references from one table to another may provide a direct binding and referential integrity may be maintained by only deleting the upper most record – that is, not leaving any dangling records. Referential integrity prevents references from being orphaned, which may lead to data loss or other more severe problems, such as a system crash. In the current embodiment, tables are stored across multiple databases. Certain tables are stored in NMS database 61 and certain other tables are stored in the configuration database within each network device in the network. Direct binding between tables cannot be maintained since a database may be removed or a record deleted without maintaining referential integrity. To address this issue, group names are used to provide a "dynamic binding" between the User Resource Group table 1018 (Fig. 11w) in the NMS database and the Managed Resource Group table 1008 (Fig. 11n) in each configuration database. Since there is no direct binding, if a group name is not found in the Managed Resource Group table, the NMS server simply returns an empty set and no data is lost or other more serious problems caused. If the group name is later added to the Managed Resource Group table, then through dynamic binding, it will be found.

Clean Copy of Paragraph 2 on Page 102

Through a user profile, a user may log-on to the network with a single, secure username and password through any NMS client, access any network device in their user profile and access configured resources corresponding to groups in their user profile. Since the tables including the data necessary for the creation of user profile LMOs are stored in the NMS database, any NMS server capable of connecting to the NMS database – that is, any NMS server in the network -- may access the tables and generate a user LMO. As a result, users may log-on with a single, secure username and password through any NMS client that may be connected to an NMS server capable of connecting to the NMS database. Essentially, users may log on through any computer system / workstation (e.g., 984, Fig. 11y) on which an NMS client is loaded or remotely through internet web access to an NMS client within the network and gain access to the network devices listed in their user profile. Thus, each user need only remember a single username and password to configure / manage any of the network devices listed in their user profile or any of the resources included within groups listed in their user profile through any NMS client in the network.

Clean Copy of Paragraph 3 beginning on Page 102 and ending on Page 103

In addition, user profiles provide a level of indirection to better protect the passwords used to access each network device. For example, access to the passwords may be limited to only those users capable of adding network devices to the network, for example, users with the administrator group access level. Other users would not see the passwords since they are automatically added to their user profile LMO, which is not accessible by users. The level of indirection provided by user profiles also allows network device passwords to be easily changed across the entire network. Periodically the passwords for access to the network devices in a network may be changed for security. The network device passwords may be quickly changed in the Administration Managed Device table 1014 (Fig. 11t), and due to the use of profiles, each user does not need to be notified of the password changes. The new passwords will be utilized automatically each time users log in. This provides for increased scalability since thousands of users will not need to be notified of the new passwords. Moreover, if a rogue user is identified, they can be quickly prevented from further access to the network through any NMS client by simply changing the user's username and/or password in the user's profile or by deleting the user's profile. Changing the username and/or password in the user profile would cause the NMS server to change the data in user table 1010 (Fig. 11r), and deleting a user profile would cause the NMS server to remove the corresponding row in the User table. In either case, the user would no longer be able to log in.

Clean Copy of Paragraph 4 beginning on Page 109 and ending on Page 110

In order to understand the significance of the PMD file (i.e., metadata), note that the MCD software does not have knowledge of card types built into it. Instead, the MCD parameterizes its operations on a particular card by looking up the card type and version number in the PMD file and acting accordingly. Consequently, the MCD software does not need to be modified, rebuilt, tested and distributed with new hardware. The changes required in the software system infrastructure to support new hardware are simpler, modify logical model 280 (Figs. 3a-3b) to include: a new entry in the PMD file (or a new PMD file) and, where necessary, new device drivers and applications. Because the MCD software, which resides in the kernel, will not need to be modified, the new applications and device drivers and the new DDL files (reflecting the new PMD file) for the configuration database and NMS database are downloaded and upgraded (as described below) without re-booting the computer system (hot upgrade).

Clean Copy of Paragraph 2 on Page 112

Referring to Fig. 13c, in one embodiment, the management subsystem 412 (Fig. 13b) is broken into two pieces: a usage data server (UDS) 412a and a file transfer protocol (FTP) client 412b. The UDS is executed on internal processor control card 542a (see also Figs. 41b and 42a-42b) while the FTP client is executed on external processor control card 542b (see also Figs. 41a and 42a-42b). Alternatively, in a network device with one processor control card or a central processor control card, both the UDS and FTP client may be executed on that one card. When each device driver, for example, SONET driver 415a-415n and ATM driver 417a-417n (only SONET driver 415a and ATM driver 417a are shown for convenience and it is to be understood that multiple drivers may be present on each card), within network device 540 is built, it links in a usage data monitoring library (UDML).

Clean Copy of Paragraph 3 beginning on Page 130 and ending on Page 131

Templates are metadata and include scripts of instructions and parameters. In one embodiment, instructions within templates are written in ASCII text to be human readable. There are three general categories of templates, provisioning templates, control templates and batch templates. A user may interactively connect the OSS client with a particular NMS server and then cause the NMS server to connect to a particular device. Instead, the user may create a control template that non-interactively establishes these connections. Once the connections are established, whether interactively or non-interactively, provisioning templates may be used to complete particular provisioning tasks. The instructions within a provisioning template cause the OSS client to issue appropriate calls to the NMS server which cause the NMS server to complete the provisioning task, for example, by writing / modifying data within the network device's configuration database. Batch templates may be used to concatenate a series of templates and template modifications (i.e., one or more control and provisioning templates) to provision one or more network devices. Through the client / server based architecture, multiple OSS clients may work with one or more NMS servers. Database view ids and APIs for the OSS client may be generated using the logical model and code generation system (Fig. 3c) to synchronize the integration interfaces between the OSS clients and the NMS servers.

Clean Copy of Paragraph 4 beginning on Page 132 and ending on Page 133

Referring to Fig. 3i, using the interactive interpreter, a network administrator may provision services by selecting (step 888) a template and using the default parameters within that template or copying and renaming (step 889) a particular provisioning template corresponding to a particular provisioning task and either accepting default parameter values provided by the template or changing (step 890) those default values to meet the administrator's needs. The network administrator may also change parameters and instructions within a copy of a template to create a new template. The modified provisioning templates are sent to or loaded into (step 891) the OSS client, which executes the instructions within the template and issues the appropriate calls (step 892) to the NMS server to satisfy the provisioning need. The OSS client may be written in JAVA and employ script technology. In response to calls received from the OSS client, the NMS server may execute (step 894) the provisioning requests defined by a template immediately or in a "batch-mode" (step 893), perhaps with other calls received from the OSS client or other clients, at a time when network transactions are typically low (e.g., late at night).

Clean Copy of Paragraph 2 on Page 133

Referring to Figs. 3j-3k, at the interactive interpreter prompt 912 (e.g., Enetcli>) a network manager may type in "help" and be provided with a list (e.g., list 913) of commands that are available. In one embodiment, available commands may include bye, close, execute, help, load, manage, open, quit, showCurrent, showTemplate, set, status, writeCurrent, and writeTemplate. Many different commands are possible. The bye command allows the network manager to exit the interactive interpreter, the close command allows the network manager to close a connection between the OSS client and that NMS server, and the execute command followed by a template type causes the OSS client to execute the instructions within the loaded template corresponding to that template type.

Clean Copy of Paragraph 2 on Page 134

The status command 918 will cause the interactive interpreter to display a status of the current interactive interpreter session. For example, the interactive interpreter may display the name 919 of an NMS server to which the OSS client is currently connected (as shown in Figs. 3j-3k, the OSS client is currently not connected to an NMS server) and the interactive interpreter may display the names 920 of available template types. The writeCurrent command followed by a template type and a new template name will cause the interactive interpreter to make a copy of the loaded template, including current parameter values, with the new template name. The writeTemplate command followed by a template type and a new template name, will cause the interactive interpreter to make a copy of the template with the new template name with placeholders values (i.e., <String>) that indicate the network manager needs to fill in the template with the required datatypes as parameter values. The network manager may then use the load command followed by the new template name to load the new template into the OSS client.

Clean Copy of Paragraph 3 on Page 134

Referring to Fig. 31, from the interactive interpreter prompt (e.g., Enetcli>), a network manager may interactively provision services on a network device. The network manager begins by typing an open command 921a followed by the IP address of an NMS server to cause the OSS client to open a connection 921b with that NMS server. The network manager may then issue a manage command 921c followed by the IP address of a particular network device to cause the OSS client to issue a call 921d to the NMS server to cause the NMS server to open a connection 921e with that network device.

Clean Copy of Paragraph 2 on Page 136

Instead of interactively establishing connections with NMS servers and network devices, control templates may be used to non-interactively establish these connections. Referring to Fig. 3m, using a showCurrent command 922 followed by CONTROL causes the interactive interpreter to display parameters available in the loaded CONTROL template. In one embodiment, an execute control command will automatically cause the OSS client to execute instructions within the loaded CONTROL template and open a connection to an NMS server designated within the CONTROL template. Since the OSS client automatically opens a connection with the designated NMS server, the open command may but need not be included within the CONTROL template. In this example, the CONTROL template includes "localhost" 923a as the DNS name of the NMS server with which the OSS client should open a connection. In one embodiment, "localhost" refers to the same system as the OSS client. A username 923b and password 923c may also need to be used to open the connection with the localhost NMS server. The CONTROL template also includes the manage command 923d and a network device IP address 923e of 192.168.9.202. With this information (and perhaps the username and password or another username and password), the OSS client issues calls to the localhost NMS server to cause the server to set up a connection with that network device.

Clean Copy of Paragraph 3 on Page 137

Referring to Fig. 3n, a batch template type named BATCH 924 includes an ordered list of tasks, including execute commands followed by provisioning template types. When a network manager issues an execute command followed by the BATCH template type at the Enetcli> prompt, the OSS client will carry out each of the tasks within the loaded BATCH template. In this example, task1 924a includes "execute SPATH" which causes the OSS client to establish a SONET path within the network device to which a connection is open, task2 924b includes "execute PVC" to cause the OSS client to set up a permanent virtual circuit within the network device, and task3 924c includes "execute SPVC" to cause the OSS client to set up a soft permanent virtual circuit within the network device.

Clean Copy of Paragraph 3 beginning on Page 138 and ending on Page 139

Batch templates may also be used to non-interactively provision services within multiple different network devices by ordering and sequencing tasks including execute commands followed by control template types and then execute commands followed by provisioning template types. Referring to Fig. 3o, instead of non-interactively establishing connections with an NMS server and a network device using a control template, a batch template may be used. For example, the first task in a loaded BATCH template 925 may be task1 925a "execute CONTROL". This will cause the OSS client to execute the loaded CONTROL template to establish connections with the NMS server and the network device designated within the loaded CONTROL template (e.g., localhost and 192.168.9.202). The BATCH template then includes provisioning tasks, for example, task2 925b includes "execute SPATH" to set up a SONET path, and task3 925c includes "set SPATH PortID 3" and task4 925d includes "execute SPATH" to set up a different SONET path. Many additional provisioning tasks for this network device may be completed in this way.

Clean Copy of Paragraph 3 beginning on Page 150 and ending on Page 151

Logical Model Change:

Where software components, including applications, device drivers, modular system services, new mission kernel images (MKIs) and diagnostic software, for a new hardware module (e.g., a line card) are not already loaded and/or if changes or upgrades (hereinafter "upgrades") to already loaded software components are needed, logical model 280 (Figs. 3a-3c) must be changed and new view ids and APIs, NMS JAVA interface files, persistent layer metadata files and new DDL files may need to be re-generated. Software model 286 is changed to include models of the new or upgraded software, and hardware model 284 is changed to include models of any new hardware. New logical model 280' is then used by code generation system 336 to re-generate view ids and APIs for any changed software components, including any new applications, for example, ATM version two 360, or device drivers, for example, device driver 362, and, where necessary, to re-generate DDL files 344' and 348' including new SQL commands and data relevant to the new hardware and/or software. The new logical model is also used to generate, where necessary, new NMS JAVA interface files 347' and new persistent layer metadata files 349'.

Clean Copy of Paragraph 2 on Page 151

Each executable software component is then built. As described above with reference to Fig. 3e, the build process involves compiling one or more source code files for the software component and then linking the resulting object code with the object code of associated libraries, a view id, an API, etc. to form an executable file. Each of the executable files and data files, for example, persistent layer metadata files and DDL files, are then provided to Kit Builder (861, Fig. 3f), which combines the components into a Network Device Installation Kit. As previously mentioned, the Kit Builder may compress each of the software components to save space. Each Installation Kit is assigned a Global release version number to distinguish between different Installation Kits.

Clean Copy of Paragraph 4 on Page 151

Software Component Signatures:

To facilitate upgrades of software components while the network device (e.g., 10, Fig. 1; 540, Figs. 35a-35b) is running (hot upgrades), a "signature" is generated for each software component. After installation (described below) within the network device of a new Installation Kit, only those software components whose signatures do not match the signatures of corresponding and currently executing software components will be upgraded. For example, different signatures associated with two ATM components represent different versions of those two ATM components.

Clean Copy of Paragraph 2 on Page 153

Referring to Fig. 20b, once a software component 1202 is built, it is passed to the signature generating program 1204, for example, the Sha-1 utility. The number generated by the signature generating program is the signature 1206 for that software component and it is appended to the built software component 1208. These steps are repeated for each software component added to the packaging list, and as Kit Builder 861 (Fig. 3f) adds each software component to the packaging list, it retrieves the signature appended to each software component and inserts it in the packaging list corresponding to the appropriate software component.

Clean Copy of Paragraph 2 on Page 161

Existing processes using their view ids and APIs to access new configuration database 42' in the same manner as they accessed old configuration database 42. However, when new processes (e.g., ATM version two 360 and device driver 362, Fig. 3c) access new configuration database 42', their view ids and APIs allow them to access new tables and data within new configuration database 42'.

Clean Copy of Paragraph 3 beginning on page 194 and ending on Page 195

Data Plane:

Referring to Figs. 35a-35b, a network device 540 includes a central processor 542, a redundant central processor 543 and a Fast Ethernet control bus 544 similar to central processors 12 and 13 and Ethernet 32 discussed above with respect to computer system 10. In addition, network device 540 includes forwarding cards (FC) 546a-546e, 548a-548e, 550a-550e and 552a-552e that are similar to line cards 16a-16n discussed above with respect to computer system 10. Network device 540 also includes (and computer system 10 may also include) universal port (UP) cards 554a-554h, 556a-556h, 558a-558h, and 560a-560h, cross-connection (XC) cards 562a-562b, 564a-564b, 566a-566b, and 568a-568b, and switch fabric (SF) cards 570a-570b. In one embodiment, network device 540 includes four quadrants where each quadrant includes five forwarding cards (e.g., 546a-546e), two cross connection cards (e.g., 562a-562b) and eight universal port cards (e.g., 554a-554h). Network device 540 is a distributed processing system. Each of the cards includes a processor and is connected to the Ethernet control bus. In addition, each of the cards are configured as described above with respect to line cards.

Clean Copy of Paragraph 3 beginning on page 196 and ending on Page 197

Referring to Figs. 36a-36b, in one embodiment, a universal port card 554a includes one or more ports 571a-571n connected to one or more transceivers 572a-572n. The user may connect an external network connection to each port. As one example, port 571a is connected to an ingress optical fiber 576a carrying an OC-48 SONET stream and an egress optical fiber 576b carrying an OC-48 SONET stream. Port 571a passes optical data from the SONET stream on fiber 576a to transceiver 572a. Transceiver 572a converts the optical data into electrical signals that it sends to a SONET framer 574a. The SONET framer organizes the data it receives from the transceiver into SONET frames. SONET framer 574a sends data over a telecommunications bus 578a to a serializer-deserializer (SERDES) 580a that serializes the data into four serial lines with twelve STS-1 time slots each and transmits the four serial lines to cross-connect card 562a.

Clean Copy of Paragraph 3 on Page 200

Multiple Redundancy Schemes:

Coupling universal port cards to forwarding cards through a cross-connection card provides flexibility in data transmission by allowing data to be transmitted from any path on any port to any port on any forwarding card. In addition, decoupling the universal port cards and the forwarding cards enables redundancy schemes (e.g., 1:1, 1+1, 1:N, no redundancy) to be set up separately for the forwarding cards and universal port cards. The same redundancy scheme may be set up for both or they may be different. As described above, the LID to PID card and port tables are used to setup the various redundancy schemes for the line cards (forwarding or universal port cards) and ports. Network devices often implement industry standard redundancy schemes, such as those defined by the Automatic Protection Switching (APS) standard. In network device 540 (Figs. 35a-35b), an APS standard redundancy scheme may be implemented for the universal port cards while another redundancy scheme is implemented for the forwarding cards.

Clean Copy of Paragraph 1 on page 201

Referring again to Figs. 35a-35b, further data transmission flexibility may be provided by connecting (i.e., connections 565) each cross-connection card 562a-562b, 564a-564b, 566a-566b and 568a-568b to each of the other cross-connection cards. Through connections 565, a cross-connection card (e.g., cross-connection card 562a) may transmit data between any port or any path on any port on a universal port card (e.g., universal port cards 554a-554h) in its quadrant to a cross-connection card (e.g., cross-connection card 568a) in any other quadrant, and that cross-connection card (e.g., cross-connection card 568a) may transmit the data to any forwarding card (e.g., forwarding cards 552a-552e) or universal port card (e.g., universal port cards 560a-560h) in its quadrant. Similarly, any cross-connection card may transmit data received from any forwarding card in its quadrant to any other cross-connection card and that cross-connection card may transmit the data to any universal port card port in its quadrant.

Clean Copy of Paragraph 4 beginning on page 205 and ending on page 206

As an example, a user connects SONET optical fiber 576a (Figs. 36a-36b) to port 571a on universal port card 554a and wants to enable a path with three time slots (i.e., STS-3c). The NMS assigns a path LID number (e.g., path LID 1666) and fills in a record (e.g., row 602) in Path Table 600 to include path LID 1666, a universal port card port LID (e.g., UP port LID 1231) previously assigned by the NMS and retrieved from the Logical to Physical Port Table, the first time slot (e.g., time slot 4) in the SONET stream corresponding with the path and the total number of time slots – in this example, 3 -- in the path. Other information may also be filled into Path Table 600.

Clean Copy of Paragraph 3 on Page 208

Multi-Layer Network Device in One Telco Rack:

Referring again to Figs. 35a-35b, in one embodiment, each universal port card includes four ports, each of which is capable of being connected to an OC-48 SONET fiber. Since an OC-48 SONET fiber is capable of transferring data at 2.5 Giga bits per second (Gbps), each universal port card is capable of transferring data at 10 Gbps (4x2.5=10). With eight port cards per quadrant, the cross-connection card must be capable of transferring data at 80 Gbps. Typically, however, the eight port cards will be 1:1 redundant and only transfer 40 Gbps. In one embodiment, each forwarding card is capable of transferring 10 Gbps, and with five forwarding cards per quadrant, the switch fabric cards must be capable of transferring data at 200 Gbps. Typically, however, the five forwarding cards will be 1:N redundant and only transfer data at 40 Gbps. With four quadrants and full redundancy (1:1 for port cards and 1:N for forwarding cards), network device 540 is capable of transferring data at 160 Gbps.

Clean Copy of Paragraph 3 beginning on page 209 and ending on page 210

To fit network device 540 into a single telco rack, dual mid-planes are used. All of the functional printed circuit boards connect to at least one of the mid-planes, and the switch fabric cards and certain control cards connect to both mid-planes thereby providing connections between the two mid-planes. In addition, to efficiently utilize routing resources, instead of providing a single cross-connection card, the cross-connection functionality is separated into four cross-connection cards – one for each quadrant – (as shown in Figs. 35a-35b). Further, routing through the lower mid-plane is improved by flipping the forwarding cards and cross-connection cards in the bottom half of the front of the chassis upside down to be the mirror image of the forwarding cards and cross-connection cards in the top of the front half of the chassis.

Clean Copy of Paragraph 3 on Page 210

The chassis also supports switch fabric cards 570a and 570b. As shown, each switch fabric card may include multiple switch fabric (SF) cards and a switch scheduler (SS) card. In addition, the chassis supports multiple central processor cards (542 and 543, Figs. 35a-35b). Instead of having a single central processor card, the external control functions and the internal control functions may be separated onto different cards as described in U.S. Patent Application Serial Number 09/574,343, filed May 20, 2000 and entitled "Functional Separation of Internal and External Controls in Network Devices", which is hereby incorporated herein by reference. As shown, the chassis may support internal control (IC) processor cards 542a and 543a and external control (EC) processor cards 542b and 543b. Auxiliary processor (AP) cards 542c and 543c are provided for future expansion to allow more external control cards to be added, for example, to handle new upper layer protocols. In addition, a management interface (MI) card 621 for connecting to an external network management system (62, Figs. 35a-35b) is also provided.

Clean Copy of Paragraph 4 beginning on page 212 and ending on Page 213

Referring to Figs. 42a-42b, mid-plane 622a includes connectors 638 mounted on the back side of the mid-plane ("back mounted") for the management interface card, connectors 640a-640d mounted on the front side of the mid-plane ("front mounted") for the quadrant 1 and 2 cross-connection cards, and front mounted connectors 642a-642b for the external control processor cards. Multiple connectors may be used for each card. Mid-plane 622a also includes back mounted connectors 644a-644p for the quadrant 1 and 2 universal port cards and front mounted connectors 646a-646j for the quadrant 1 and 2 forwarding cards.

Clean Copy of Paragraph 1 on Page 215

Referring again to Figs. 36a-36b, as described above, each forwarding card (e.g., 546c) includes traffic management chips (e.g., 588a-588n and 590a-590b) that ensure high priority network data / traffic (e.g., voice) is transferred faster than lower priority traffic (e.g., e-mail). Each forwarding card also includes switch fabric interface (SFIF) chips (e.g., 589a-589n) that transfer network data between the traffic management chips and the switch fabric cards 570a-570b.

Clean Copy of Paragraph 2 on Page 215

Referring also to Fig. 43, forwarding card 546c includes traffic management (TM) chips 588n and 590a and SFIF chips 589, and forwarding card 550a includes traffic management chips 659a and 659b and SFIF chips 660. (Fig. 43 includes only two forwarding cards for convenience but it is to be understood that many forwarding cards may be included in a network device as shown in Figs. 35a-35b.) SFIF chips 589 and 660 on both boards include a switch fabric interface (SIF) chip 661, data slice chips 662a-662f, an enhanced port processor (EPP) chip 664 and a local timing subsystem (LTS) 665. The SFIF chips receive data from ingress TM chips 588n and 659a and forward it to the switch fabric cards 570a – 570b (Figs. 36a-36b). Similarly, the SFIF chips receive data from the switch fabric cards and forward it to the egress TM chips 590a and 659b.

Clean Copy of Paragraph 2 on Page 221

As previously mentioned, the network device may include redundant switch fabric cards 570a and 570b (Figs. 36a-36b) and as described above with reference to Fig. 43, each switch fabric card 570a and 570b may include a control card and four or more data cards. Referring to Fig. 44, network device 540 may include switch fabric control card 666 (part of central switch fabric 570a) and redundant switch fabric control card 667 (part of redundant switch fabric 570b). Each control card 666 and 667 includes a central timing subsystem (CTS) 673. One CTS behaves as the master and the other CTS behaves as a slave and locks its output SOS signal to the master's output SOS signal. In one embodiment, upon power-up or system re-boot the CTS on the primary switch fabric control card 666 begins as the master and if a problem occurs with the CTS on the primary control card, then the CTS on redundant control card 667 takes over as master without requiring a switch over of the primary switch fabric control card.

Clean Copy of Paragraph 3 on Page 222

Central Timing Subsystem (CTS):

Referring to Figs. 45a-45b, central timing subsystem (CTS) 673 includes a voltage controlled crystal oscillator (VCXO) 676 that generates a 25MHz reference SOS signal 678. The SOS signal must be distributed to each of the local timing subsystems (LTSs) and is, thus, sent to a first level clock driver 680 and then to second level clock drivers 682a-682d that output reference SOS signals SFC_BENCH_FB and SFC_REF1 – SFC_REFn. SFC_BENCH_FB is a local feedback signal returned to the input of the CTS. One of SFC_REF1 - SFC_REFn is sent to each LTS, the other CTS, which receives it on SFC_SYNC, and one is routed over a mid-plane and returned as a feedback signal SFC_FB to the input of the CTS that generated it. Additional levels of clock drivers may be added as the number of necessary reference SOS signals increases.

Clean Copy of Paragraph 3 beginning on page 223 and ending on Page 224

When the CTS is the slave, hardware control logic 684 selects slave VCXO voltage signal 688b. This provides a variable voltage level to the VCXO that causes the output of the VCXO to track or follow the SOS reference signal from the master CTS. Referring still to Figs. 45a-45b, the CTS receives the SOS reference signal from the other CTS on SFC_SYNC. Since this is a differential PECL signal, it is first passed through a differential PECL to TTL translator 692 before being sent to MUX 697a within dual MUX 694. In addition, two feedback signals from the CTS itself are supplied as inputs to the CTS. The first feedback signal SFC_FB is an output signal (e.g., one of SFC_REF1-SFC_REFn) from the CTS itself which has been sent out to the mid-plane and routed back to the switch fabric control card. This is done so that the feedback signal used by the CTS experiences identical conditions as the reference SOS signal delivered to the LTSs and skew is minimized. The second feedback signal SFC_BENCH_FB is a local signal from the output of the CTS, for example, clock driver 682a. SFC_BENCH_FB may be used as the feedback signal in a test mode, for example, when the control card is not plugged into the network device chassis and SFC_SB is unavailable. SFC_BENCH_FB and SFC_FB are also differential PECL signals and must be sent through translators 693 and 692, respectively, prior to being sent to MUX 697b within dual MUX 694. Hardware control logic 684 selects which inputs are used by MUX 694 by asserting signals on REF_SEL(1:0) and FB_SEL(1:0). In regular use, inputs 696a and 696b from translator 692 are selected. In test modes, grounded inputs 695a, test headers 695b or local feedback signal 698 from translator 693 may be selected. Also in regular use (and in test modes where a clock signal is not inserted through the test headers), copies of the selected input signals are provided on the test headers.

Clean Copy of Paragraph 3 beginning on Page 228 and ending on Page 229

Master / Slave CTS Control:

In order to determine which CTS is the master and which is the slave, hardware control logic 684 implements a state machine. Each hardware control logic 684 sends an IM_THE_MASTER signal to the other hardware control logic 684 which is received as a YOU_THE_MASTER signal. If the IM_THE_MASTER signal – and, hence, the received YOU_THE_MASTER signal -- is asserted then the CTS sending the signal is the master (and selects input 688a to MUX 686, Figs. 45a-45b) and the CTS receiving the signal is the slave (and selects input 688b to MUX 686). Each IM_THE_MASTER / YOU_THE_MASTER etch is pulled down to ground on the mid-planes such that if one of the CTSs is missing, the YOU_THE_MASTER signal received by the other CTS will be a logic 0 causing the receiving CTS to become the master. This situation may arise, for example, if a redundant control card including the CTS is not inserted within the network device. In addition, each of the hardware control logics receive SLOT_ID signals from pull-down/pull-up resistors on the chassis mid-plane indicating the slot in which the switch fabric control card is inserted.

Clean Copy of Paragraph 2 on Page 230

Local Timing Subsystem:

Referring to Figs. 47a-47b, each local timing subsystem (LTS) 665 receives a reference SOS signal from each CTS on SFC_REFA and SFC_REFB. Since these are differential PECL signals, each is passed through a differential PECL to TTL translator 714a or 714b, respectively. A feedback signal SFC_FB is also passed from the LTS output to both translators 714a and 714b. The reference signal outputs 716a and 716b are fed into a first MUX 717 within dual MUX 718, and the feedback signal outputs 719a and 719b are fed into a second MUX 720 within dual MUX 718. LTS hardware control logic 712 controls selector inputs REF_SEL (1:0) and FB_SEL (1:0) to dual MUX 718. With regard to the feedback signals, the LTS hardware control logic selects the feedback signal that went through the same translator as the reference signal that is selected to minimize the effects of any skew introduced by the two translators.

Clean Copy of Paragraph 3 beginning on page 233 and ending on Page 234

To accomplish this, a latch 547 (Fig. 40) on the faceplate of each of the printed circuit boards on which a distributed switch fabric is located is connected to a circuit 742 (Fig. 44) also on the printed circuit board that detects when the latch is released. When the latch is released, indicating that the board is going to be removed from the network device, circuit 742 sends a signal to a circuit 743 on both switch fabric control cards indicating that the forwarding card is about to be removed. Circuit 743 sends an interrupt to the local processor (e.g., 710, Figs. 45a-45b) on the switch fabric control card. Software (e.g., slave SRM) being executed by the local processor detects the interrupt and sends a notice to software (e.g., master SRM) being executed by the processor (e.g., 24, Fig. 1) on the network device centralized processor card (e.g., 12, Fig. 1, 542 or 543, Figs. 35a-35b). The master SRM sends a notice to the slave SRMs being executed by the processors on the switch fabric data cards and forwarding cards to indicate the removal of the forwarding card. The redundant forwarding card switches over to become a replacement for the failed primary forwarding card. The master SRM also sends a notice to the slave SRM on the cross-connection card (e.g., 562-562b, 564a-564b, 566a-566b, 568a-565b, Figs. 35a-35b) to re-configure the connections between the port cards (e.g., 554a-554h, 556a-556h, 558a-558h, 560a-560h, Figs. 35a-35b) and the redundant forwarding card. The slave SRM on the switch fabric control card re-configures the registers in the scheduler component to disable the scheduler's links to the EPP chip on the forwarding card that's being removed from the network device. As a result, when the forwarding card is removed, the scheduler will not detect an error due to a missing EPP chip.

Clean Copy of Paragraph 3 on Page 235

To limit the amount of time that data transfer is stopped in a network device including distributed switch fabric subsystems, the local processors each set up for a refresh while data is still being transferred. Communications between the processors take place over the Ethernet bus (e.g., 32, Fig. 1, 544, Figs. 35a-35b) to avoid interrupting network data transfer. When all processors have indicated (over the Ethernet bus) that they are ready for the refresh, the processor on the master switch fabric control card stops data transfer and sends a refresh command to each of the processors on the forwarding cards and switch fabric cards. Since all processors are waiting to complete the refresh, it is quickly completed. Each processor notifies the processor on the master switch fabric control card that the refresh is complete, and when all processors have completed the refresh, the master switch fabric control card re-starts the data transfer.

Clean Copy of Paragraph 2 on Page 237

Referring to Fig. 49, controller card 542b and redundant controller card 543b each include an external central timing subsystem (EX CTS) 750. Each EX CTS receives BITS lines 751 and provide BITS lines 752. In addition, each EX CTS receives a port timing signal 753 from each port card (554a-554h, 556a-556h, 558a-558h, 560a-560h, Figs. 35a-35b), and each EX CTS also receives an external timing reference signal 754 from itself and an external timing reference signal 755 from the other EX CTS.

Clean Copy of Paragraph 4 on Page 237

An external reference timing signal from each EX CTS is sent to each external local timing subsystem (EX LTS) 756 on cards throughout the network device, and each EX LTS generates local external timing signals synchronized to one of the received external reference timing signals. Generally, external reference timing signals are sent only to cards including external data transfer functionality, for example, cross connection cards 562a-562b, 564a-564b, 566a-566b and 568a-568b (Figs. 35a-35b) and universal port cards 554a-554h, 556a-556h, 558a-558h, 560a-560h.

Clean Copy of Paragraph 2 on Page 239

External Central Timing Subsystem (EX CTS):

Referring to Figs. 50a-50c, EX CTS 750 includes a T1/E1 framer/LIU 758 for receiving and terminating BITS signals 751 and for generating and sending BITS signals 752. Although T1/E1 framer is shown in two separate boxes in Figs. 50a-50c, it is for convenience only and may be the same circuit or component. In one embodiment, two 5431 T1/E1 Framer Line Interface Units (LIU) available from PMC-Sierra are used. The T1/E1 framer supplies 8KHz BITS_REF0 and BITS_REF1 signals and receives 8KHz BITS1_TXREF and BITS2_TXREF signals. A network administrator notifies NMS 60 (Figs. 35a-35b) as to whether the BITS signals are T1 or E1, and the NMS notifies software running on the network device. Through signals 761 from a local processor, hardware control logic 760 within the EX CTS is configured for T1 or E1 and sends an T1E1_MODE signal to the T1/E1 framer indicating T1 or E1 mode. The T1/E1 framer then forwards BITS_REF0 and BITS_REF1 to dual MUXs 762a and 762b.

Clean Copy of Paragraph 1 on Page 243

Similar to hardware control logic 684 (Figs. 45a-45b) within the switch fabric CTS, hardware control logic 760 within the EX CTS implements a state machine (similar to the state machine shown in Fig. 46) based on IM_THE_MASTER and YOU_THE_MASTER signals sent between the two EX CTSs and also on slot identification signals (not shown).

Clean Copy of Paragraph 1 on Page 246

External Local Timing Subsystem (EX LTS):

Referring to Figs. 54a-54b, EX LTS 756 receives STRAT_REF_B from one EX CTS and STRAT_REF_A from the other EX CTS. STRAT_REF_B and STRAT_REF_A correspond to one of STRAT_REF1-STRAT_REFN (Figs. 50a-50c) output from each EX CTS. STRAT_REF_B and STRAT_REF_A are provided as inputs to a MUX 810a and a hardware control logic 812 within the EX LTS selects the input to MUX 810a using REF_SEL (1:0) signals. An activity detector 814a monitors the activity of STRAT_REF_A and sends a signal to hardware control logic 812 if it detects an inactive signal (i.e., stuck at logic one or zero). Similarly, an activity detector 814b monitors the activity of STRAT_REF_B and sends a signal to hardware control logic 812 if it detects an inactive signal (i.e., stuck at logic one or zero). If the hardware control logic receives a signal from either activity detector indicating that the monitored signal is inactive, the hardware control logic automatically changes the REF_SEL (1:0) signals to cause MUX 810a to select the other input signal and send an interrupt to the local processor.

Clean Copy of Paragraph 4 on Page 247

Similar to hardware control logic 712 (Figs. 47a-47b) within the switch fabric LTS, hardware control logic 812 within the EX LTS implements a state machine (similar to the state machine shown in Fig. 48) based on signals from activity detectors 814a and 814b.

Clean Copy of Paragraph 3 on Page 248

External Central Timing Subsystem (EX CTS) Alternate Embodiment:

Referring to Figs. 55a-55c, instead of using one of the STRAT_REF1-STRAT_REFN signals from the other EX CTS as an input to MUX 772a, the output 770 (marked "Alt. Output to other EX CTS") of timing module 768 may be provided to the other EX CTS and received as input 838 (marked "Alt. Input from other EX CTS"). The PLL circuit, including MUXs 772a and 772b, phase detector 776, discrete logic circuit 778 and VCXO 780, is necessary to synchronize the output of the VCXO with either output 770 of the timing module or a signal from the other EX CTS. However, PLL circuits may introduce jitter into their output signals (e.g., output 781), and passing the PLL output signal 781 via one of the STRAT_REF1-STRAT_REFN signals from one EX CTS into the PLL of the other EX CTS – that is, PLL to PLL -- may introduce additional jitter into output signal 781. Since accurate timing signals are critical for proper data transfer with other network devices and SONET standards specifically set maximum allowable jitter transmission at interfaces (Bellcore GR-253-CORE and SONET Transport Systems Common Carrier Criteria), jitter should be minimized. Passing the output 770 of the timing module within the EX CTS to the input 838 of the other EX CTS avoids passing the output of one PLL to the input of the second PLL and thereby reduces the potential introduction of jitter.

Clean Copy of Paragraph 2 on page 249

Layer One Test Port:

The present invention provides programmable physical layer (i.e., layer one) test ports within an upper layer network device (e.g., network device 540, Figs. 35a-35b). The test ports may be connected to external test equipment (e.g., an analyzer) to passively monitor data being received by and transmitted from the network device or to actively drive data to the network device. Importantly, data provided at a test port accurately reflects data received by or transmitted by the network device with minimal modification and no upper layer translation or processing. Moreover, data is supplied to the test ports without disrupting or slowing the service provided by the network device.

Clean Copy of Paragraph 3 on Page 249

Referring to Figs. 35a-35b and 36a-36b, network device 540 includes at least one cross-connection card 562a-562b, 564a-564b, 566a-566b, 568a-568b, at least one universal port card 554a-554h, 556a-556h, 558a-558h, 560a-560h, and at least one forwarding card 546a-546e, 548a-548e, 550a-550e, 552a-552e. Each port card includes at least one port 571a-571n for connecting to external physical network attachments 576a-576b, and each port card transfers data to a cross-connection card. The cross-connection card transfers data between port cards and forwarding cards and between port cards. In one embodiment, each forwarding card includes at least one port/payload extractor 582a-582n for receiving data from the cross-connection cards.

Clean Copy of Paragraph 2 on page 250

For many reasons, including error diagnosis, a service administrator may wish to monitor the data received on a particular path or paths at a particular port, for example, port 571a, and/or the data transmitted on a particular path or paths from port 571a. To accomplish this, the network administrator may connect test equipment, for example, an analyzer 840 (e.g., an Omniber analyzer available from Hewlett Packard Company), to the transmit connection of port 571b to monitor data received at port 571a and/or to the transmit connection of port 571c to monitor data transmitted from port 571a. The network administrator then notifies the NMS (e.g., NMS 60 running on PC 62, Figs. 35a-35b) as to which port or ports on which port card or port cards should be enabled and whether the transmitter and/or receiver for each port should be enabled. The network administrator also notifies the NMS as to which path or paths are to be sent to each test port, and the time slot for each path. With this information, the NMS fills in test path table 841 (Figs. 57 and 58) in configuration database 42.

Clean Copy of Paragraph 3 on Page 254

NMS Server Scalability

As described above, a network device (e.g., 10, Fig. 1 and 540, Figs. 35a-35b) may include a large number (e.g., millions) of configurable / manageable objects. Manageable objects are typically considered physical or logical. Physical managed objects correspond to the physical components of the network device such as the network device itself, one or more chassis within the network device, shelves in each chassis, slots in each shelf, cards inserted in each slot, physical ports on particular cards (e.g., universal port cards), etc. Logical managed objects correspond to configured elements of the network device such as SONET paths, internal logical ports (e.g., forwarding card ports), ATM interfaces, virtual ATM interfaces, virtual connections, paths/interfaces related to other network protocols (e.g., MPLS, IP, Frame Relay, Ethernet), etc.

Clean Copy of Paragraph 2 on Page 255

Referring to Fig. 59, an NMS client 850a runs on a personal computer or workstation 984 and uses data in graphical user interface (GUI) tables 985 stored in local memory 986 to display a GUI to a user (e.g., network administrator, provisioner, customer) after the user has logged in. In one embodiment, the GUI is GUI 895 described above with reference to Figs. 4a-4z, 5a-5z, 6a-6p, 7a-7y, 8a-8e, 9a-9n, 10a-10i and 11a-11h. When GUI 895 is initially displayed (see Fig. 4a), only navigation tree 898 is displayed and under Device branch 898á a list 898b of IP addresses and/or domain name server (DNS) names may be displayed corresponding to network devices that may be managed by the user in accordance with the user's profile.

Clean Copy of Paragraph 2 on Page 256

In one embodiment, data is stored within configuration database 42 as a series of containers. Since the configuration database is a relational database, data is stored in tables and containment is accomplished using pointers from lower level tables (children) to upper level tables (parents). As previously discussed with reference to Figs. 12a-12c, after the network device is powered-up, the Master MCD (Master Control Driver) 38 takes a physical inventory of the network device (e.g., computer system 10, Fig. 1, network device 540, Figs. 35a-35b, 59) and assigns a unique physical identification number (PID) to each physical component within the system, including the network device itself, each chassis in the network device, each shelf in each chassis, each slot in each shelf, each card inserted in each slot, and each port on each card having a physical port (e.g., universal port cards). As previously stated, the PID is a unique logical number unrelated to any physical aspect of the component.

Clean Copy of Paragraph 2 on Page 257

Referring to Fig. 60a, since the managed device is the top physical level, managed device table 983 includes one row 983a representing the one managed device (e.g., 540, Figs. 35a-35b and 59) including a unique managed device PID 983b (e.g., 1; i.e., primary key) and attributes A1-An corresponding to the managed device but the managed device table does not include a parent PID (i.e., foreign key for association). In the current embodiment, chassis table 988 includes one row 988a representing the one chassis (e.g., 620, Figs. 41a-41b) in the managed device. Other network devices may have multiple chassis and a row would be added to the chassis table for each chassis and each row would include the same managed device PID (e.g., 1). Each row in the chassis table includes a unique chassis PID 988b (e.g., 2; i.e., primary key) and attributes A1-An corresponding to the chassis and a managed device PID 988c (i.e., parent PID / foreign key for association). Referring to Fig. 60c, shelf table 989 includes one row for each shelf in the chassis and each row includes a unique shelf PID 989a (e.g., 3-18; i.e., primary key) and attributes A1-An corresponding to each shelf and a chassis PID 989b (i.e., foreign key for association). Since all the shelves are in the same chassis in this embodiment, they each list the same chassis PID (e.g., 2). Referring to Fig. 60d, slot table 990 includes one row for each slot in the chassis and each row includes a unique slot PID 990a (e.g., 20-116; i.e., primary key) and attributes A1-An corresponding to each slot and a shelf PID 990b (i.e., foreign key for association). Since there may be many shelves in the chassis, the shelf PID in each row corresponds to the shelf in which the slot is located. For example, a row 990c includes slot PID 20 corresponding to a shelf PID of 3, and a row 990d includes slot PID 116 corresponding to a different shelf PID of 18.

Clean Copy of Paragraph 1 on Page 270

As previously discussed, each SONET path corresponds to a port (e.g., 571a, Figs. 36a-36b) on a universal port card (e.g., 554a) and is connected through a cross-connection card (e.g., 562a) to a service end point corresponding to a port (i.e., slice) on a forwarding card (e.g., 546c). In one embodiment, after filling in one or more rows in SONET Path Table 600', the NMS server also fills in one or more corresponding rows in Service EndPoint Table (SET) 76'' (Fig. 60h). The NMS server assigns a unique service endpoint LID 76a (i.e., primary key) to each service endpoint and inserts the service endpoint LID within a corresponding row. The NMS server also inserts the corresponding path LID 76b (i.e., foreign key for association) within each row and may also insert attributes associated with each service endpoint. For example, the NMS server may insert the quadrant number corresponding to the selected port and may also insert other attributes (if provided by the user) such as the forwarding card slice PID (76d) corresponding to the service end point, the forwarding card PID (76c) on which the port / slice is located and the forwarding card time slot (76e). Alternatively, the NMS server only provides the quadrant number attribute and a policy provisioning manager (PPM) 599 (Fig. 37) decides which forwarding card, slice (i.e., payload extractor chip) and time slot (i.e., port) to assign to the new universal port card path, and once decided, the PPM fills in SET Table 76'' attribute fields (i.e., self-completing configuration record).

Clean Copy of Paragraph 4 on Page 278

Network Device Authentication:

When a user selects an IP address (i.e., 192.168.9.202, Fig. 4e) representing a particular network device from device list 898b in GUI 895, a network management system (NMS) client (e.g., 850a, Fig. 2b) sends a message to an NMS server (e.g., 851a) and the NMS server uses the IP address to connect to the network device (e.g., 540) to which that IP address is assigned. The NMS server may connect to a network device port on a universal port card for in-band management or a port on an external Ethernet bus 41 (Figs. 13b and 35a-35b) for out-of-band management.

Clean Copy of Paragraph 5 beginning on Page 278 and ending on Page 279

For out-of-band management, the NMS server uses the IP address over a separate management network, typically a local area network (LAN), to reach an interface 1036 (Figs. 63a-63b) on the network device to external Ethernet bus 41. Any intermediate network may exist between the local network to which the NMS is connected and the local network (i.e., Ethernet 41) to which the network device is connected. A Media Access Control (MAC) address (hereinafter referred to as the network device's external MAC address) is then used on Ethernet 41 to bridge the packet, containing the IP address, to the network device.

Clean Copy of Paragraph 3 on Page 279

Referring to Figs. 63a-63b, in one embodiment, an external Ethernet network interface 1036 for connecting network device 540 to external Ethernet 41 is located on management interface (MI) card 621 (see also Fig. 41a), and the IEEE provided MAC address (i.e., external MAC address) assigned to the MI card is stored in PROM 1038.

Clean Copy of Paragraph 2 on Page 281

As described above, when a network device is added to a network, an administrator selects an Add Device option in a pop-up menu 898c (Fig. 6a) in GUI 895 to cause a dialog box (e.g., 898d, Fig. 6b; 1013, Fig. 11u) to be displayed. After entering the required information into the dialog box, the user selects an Add button (e.g., 898f, Fig. 6b; 1013h, Fig. 11u). Selection of the Add button causes the NMS client to send the data from the dialog box to the NMS server. The NMS server adds a row to Administration Managed Device table 1014' (Fig. 64) and inputs the data sent from the NMS client into the new row. In addition, the NMS server uses the IP address in the data sent from the NMS client to connect with the network device and retrieve two or more physical identifiers. The physical identifiers may then be stored in columns (e.g., 1014e' and 1014f') of the Administration Managed Device table. Although only two physical identifier (ID) columns are shown in Fig. 64, the Administration Managed Device table may include additional columns for additional physical identifiers.

Clean Copy of Paragraph 3 beginning on page 303 and ending on Page 304

The bus bars are used to distribute power through the midplanes to each of the modules requiring power that are plugged into connectors (see Figs. 42a-42b) on the midplanes. Bus bars 1086a and 1086b are connected with bus bars 1082a and 1082b, respectively, on the lower midplane which are connected with bus bars 1088a and 1088b, respectively, on the upper midplane 622a. Similarly, bus bars 1086e, 1086f, 1086i and 1086j are connected with bus bars 1082c, 1082d, 1082e and 1082f, respectively, on the lower midplane which are connected with bus bars 1088c, 1088d, 1088e and 1088f, respectively, on the upper midplane. The bus bars on the midplanes are connected using metal straps 1089 (Fig. 73b). Bus bars 1086c, 1086d, 1086g and 1086f are connected with etches (not shown) located on internal layers within the lower midplane which are then connected with etches (not shown) located on internal layers within the upper midplane.

Clean Copy of Paragraph 3 beginning on page 318 and ending on Page 319

Referring to Figs. 35a-35b, 41a and 41b, network device 540 includes a distributed processing system where each card, except management interface (MI) card 621, includes a processor and is connected to an internal Ethernet switch network 544. The distributed processors transmit internal control information and external control information taken from the data path to each other over the Ethernet switch (i.e., out-of-band management). Preferably, the Ethernet switch is a Fast Ethernet providing 100Mb of bandwidth to each processor to allow the control information to be exchanged at high frequencies. The external control information may be assigned a higher priority than the internal control information to insure that it is not dropped during periods of heavy traffic. A backup or redundant Ethernet switch may also be connected to each processor such that if the primary Ethernet switch fails, the cards can fail-over to the backup Ethernet switch and keep the network device running.

Clean Copy of Paragraph 2 on Page 323

Providing an internal out-of-band control path with dedicated resources for each processor eliminates or minimizes the issues related to in-band management and out-of-band management using a shared media control path but is expensive in terms of network device resources, including the cost of the components and the space required for those components and for the signals to connect those components. Each connection between a physical port within an Ethernet switch subsystem and a card requires four signals. Thus, each of the cards must dedicate four pins and the four signals must be routed across those cards and across the midplanes 622a, 622b (Figs. 42a-42b). The internal processor cards must dedicate four pins for each physical port within the Ethernet switch subsystem located thereon as well as four pins for each physical port (e.g., 1152a, 1162b) that connects to the Ethernet switch subsystem located thereon and on the other internal processor card. Where a card connects to two ports within each Ethernet switch subsystem, 16 pins must be dedicated and 16 signals routed within the card and across the midplane.

Clean Copy of Paragraph 3 beginning on Page 323 and ending on Page 324

Isolated Ethernet Switch Address Generation

As described above, typically each card including an Ethernet port is assigned a unique MAC address, which is stored in a programmable read only memory component (PROM) on the card. The MAC address for each card is a globally unique identifier for the card on any Ethernet network. As described above with reference to Figs. 63a-63b, in one embodiment, an external Ethernet port 1036 for connecting network device 540 to external Ethernet 41 is located on management interface (MI) card 621 (see also Fig. 41a), and a MAC address assigned to the MI card is stored in PROM 1038. Similarly, a MAC address may be assigned to each card in network device 540 including an Ethernet port (that is, each card including a processor) and stored in a PROM on the card.